MONASH University

FIT5195 Business Intelligence and Data Warehousing

Major Assignment

Rishabh Arora

Student ID: 31337376

C. Tasks

C.1 Design a data warehouse for the M-Stay database

a) The E/R diagram of the operational database

The below figure shows the E/R Diagram of the operational database:

b) Data Cleaning

In this section, data exploration and data cleaning has been performed.

Error 1: Duplicate values in booking table

The below code has been used to identify duplicate values in booking table.

```
select count(*) from mstay.booking;
```

	COUNT(*)
1	5002

```
select count(*) from
(select distinct * from mstay.booking);
```

	COUNT(*)
1	5001

```
select booking_id, count(*)
from mstay.booking
group by booking_id
having count(*)>1;
```

	BOOKING_ID	COUNT(*)
1	537	2

The above query and outputs show that booking ID - 537 has a duplicate record. The below query is used to clean the data.

```
create table booking as
(select distinct * from mstay.booking);
```

The below query is used to verify if the duplicate value has been removed.

select count(*) from booking;

	COUNT(*)
1	5001

select count(*) from

*(select distinct * from booking);*

	COUNT(*)
1	5001

The above outputs show that duplicate value has been removed.

Error 2: Duplicate values in host table

The below query has been used to identify duplicate values in host table.

select count(*) from mstay.host;

	COUNT(*)
1	3883

select count(*) from

*(select distinct * from mstay.host);*

	COUNT(*)
1	3880

select host_id, count(*)

from mstay.host

group by host_id

having count(*)>1;

	HOST_ID	COUNT(*)
1	7046664	4

The above query and outputs show that host ID - 7046664 has a duplicate record. The below code is used to clean the data.

create table host as

*(select distinct * from mstay.host);*

The below query is used to verify if the duplicate value has been removed.

select count(*) from host;

	COUNT(*)
1	3880

select count(*) from

*(select distinct * from host);*

	COUNT(*)
1	3880

The above outputs show that duplicate value has been removed.

Error 3: Invalid booking id found in review table

The below query has been used to identify invalid values in review table.

*select * from mstay.review*

where booking_id not in (select booking_id from mstay.booking);

	REVIEW_ID	REVIEW_DATE	REVIEW_COMMENT	BOOKING_ID
1	734	19/DEC/21	super location, awesome staff, our team absolutely loved this dinner location. well done to the whole team at rice, paper, scissors.	500123

The above query and outputs show that there is an invalid booking ID in the review table. The below code is used to clean the data.

create table review as

*(select * from mstay.review*

where booking_id in (select booking_id from mstay.booking));

The below query is used to verify if the invalid value has been removed.

*select * from review*

where booking_id not in (select booking_id from mstay.booking);

The above outputs show that invalid values have been removed.

Error 4: Invalid property id and host_id found in listing table

The below query has been used to identify invalid values in listing table.

*select * from mstay.listing*

where prop_id not in (select prop_id from mstay.property);

	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_PRICE	LISTING_MIN_NIGHTS	LISTING_MAX_NIGHTS	PROP_ID	TYPE_ID	HOST_ID
1	99999	18/DEC/18	Melbourne accomodation	-150	1	7	9999	2	9999

*select * from mstay.listing*

where host_id not in (select host_id from mstay.host);

	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_PRICE	LISTING_MIN_NIGHTS	LISTING_MAX_NIGHTS	PROP_ID	TYPE_ID	HOST_ID
1	99999	18/DEC/18	Melbourne accomodation	-150	1	7	9999	2	9999

The above query and outputs show that there is an invalid property ID and host ID in the review table. The below code is used to clean the data.

create table listing as

(

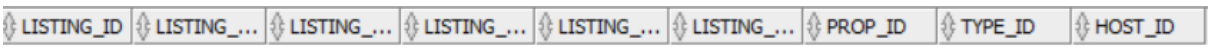
*select * from mstay.listing*

where prop_id in (select prop_id from mstay.property));

The below query is used to verify if the invalid value has been removed.

*select * from listing*

where prop_id not in (select prop_id from mstay.property);



*select * from listing*

where host_id not in (select host_id from mstay.host);

LISTING_ID	LISTING_...	LISTING_...	LISTING_...	LISTING_...	LISTING_...	PROP_ID	TYPE_ID	HOST_ID

The above outputs show that invalid values have been removed.

Error 5: Invalid channel id and host id found in host verification table

The below query has been used to identify invalid values in host verification table.

*select * from mstay.host_verification*

where channel_id not in (select channel_id from mstay.channel);

	HOST_ID	CHANNEL_ID
1	123	17

*select * from mstay.host_verification*

where host_id not in (select host_id from mstay.host);

	HOST_ID	CHANNEL_ID
1	123	17

The above query and outputs show that there is an invalid channel ID and host ID in the host_verification table. The below code is used to clean the data.

create table host_verification as

*(select * from mstay.host_verification*

where host_id in (select host_id from mstay.host));

The below query is used to verify if the invalid value has been removed.

*select * from host_verification*

where channel_id not in (select channel_id from mstay.channel);

HOST_ID	CHANNEL...

*select * from host_verification*

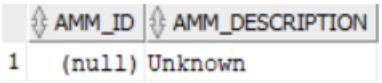
where host_id not in (select host_id from mstay.host);

The above outputs show that invalid values have been removed.

Error 6: Null values in amenity table primary key

The below query has been used to identify null values in amenity table.

*select * from mstay.amenity where amm_id is null;*



The above query and output show that there is a null value in amenity table. The below code is used to clean the data.

create table amenity as

*(select * from mstay.amenity*

where amm_id is not null);

The below query is used to verify if the null value has been removed.

*select * from amenity where amm_id is null;*

The above outputs show that null values have been removed.

Error 7: Out of bound listing price value in listing table

The below query has been used to identify out of bound values in listing table.

*select * from mstay.listing*

where listing_price<0;

	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_PRICE	LISTING_MIN_NIGHTS	LISTING_MAX_NIGHTS	PROP_ID	TYPE_ID	HOST_ID
1	99999	18/DEC/18	Melbourne accomodation	-150	1	7	9999	2	9999

The above query and output show that there is an invalid value listing price value in listing table. The below code is used to clean the data.

delete from listing where listing_price<0;

The below query is used to verify if the invalid value has been removed.

*select * from listing*

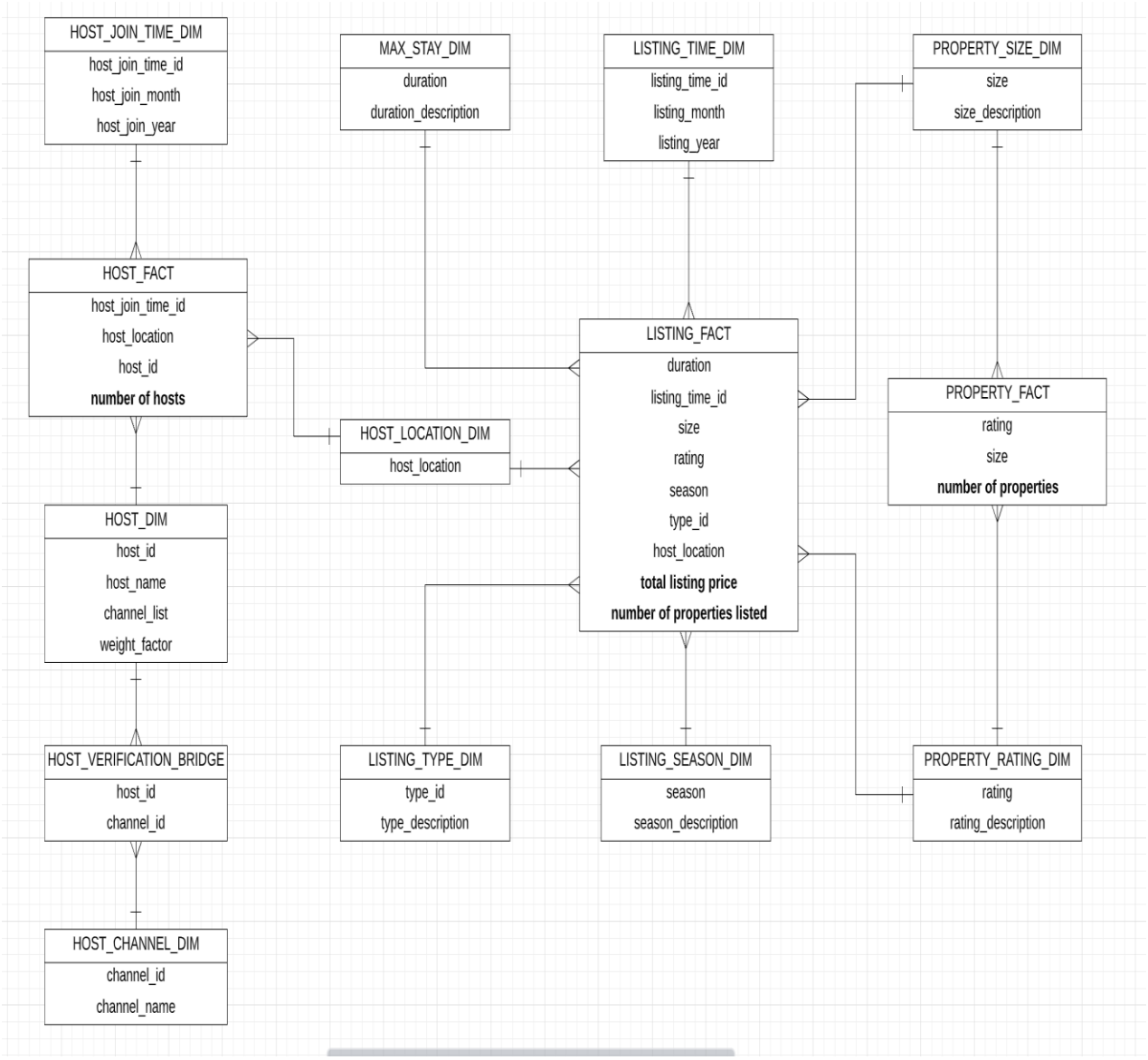
where listing_price<0;

LISTING_ID	LISTING_...	LISTING_...	LISTING_...	LISTING_...	LISTING_...	PROP_ID	TYPE_ID	HOST_ID

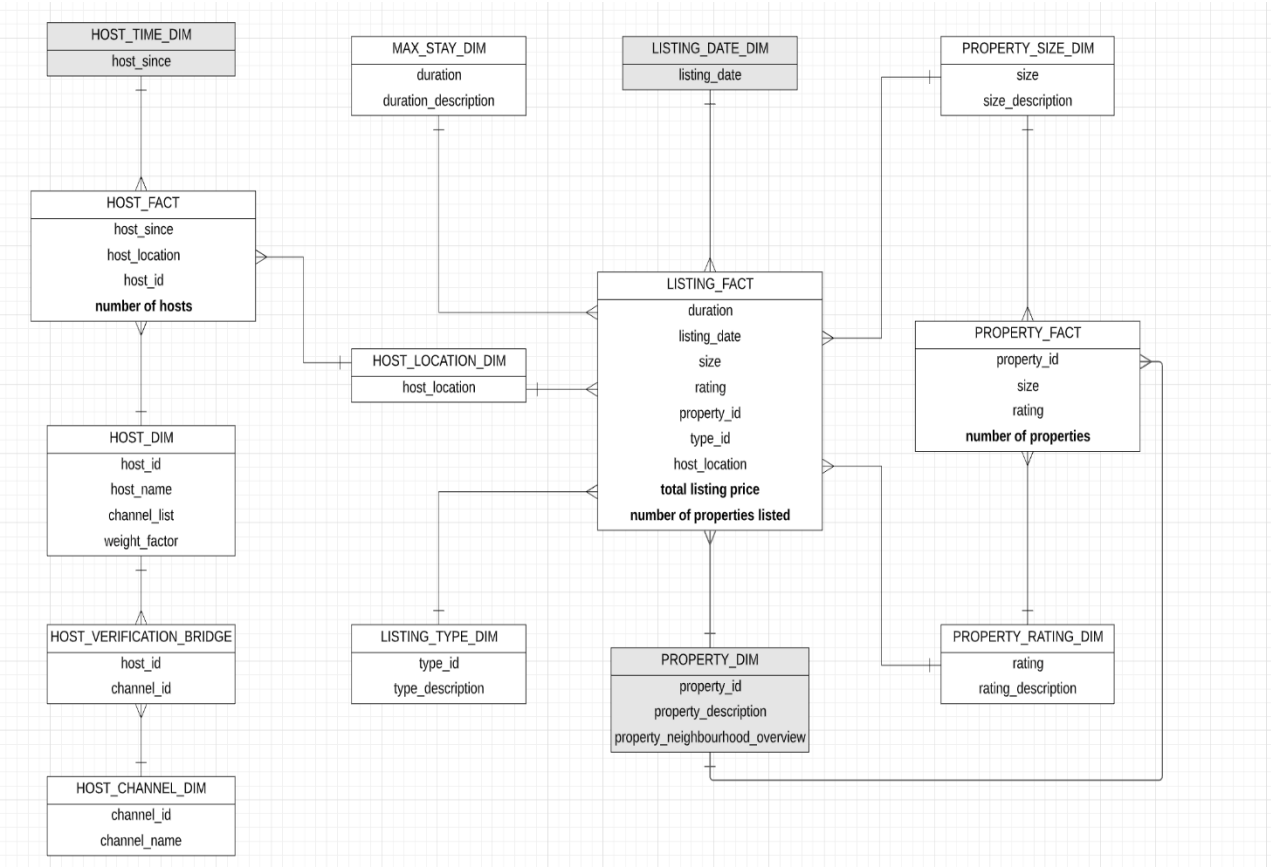
The above outputs show that invalid values have been removed.

c) Two versions of star/snowflake schema diagrams

In this section, two versions of star/snowflake diagrams are created. The first version is at the highest level of aggregation and second version is at level 0 i.e., no aggregation. The below figure shows the version 1 star schema.



The below figure shows the version 2 star/snowflake schema



d) The reasons for the choice of SCD type for temporal dimension, if any

Data Exploration was performed on the listing table to identify slowly changing dimension (SCD). The following query was used.

```
select prop_id,listing_price
from listing
where prop_id in
(select prop_id
from listing
group by prop_id
having count(*)>1);
```

On exploring the above query result, it was found that listing price of properties did not change over the time. Moreover, no listing price history table was found in the operational database. Even if the listing prices of the property would have changed, we assume the database stores the latest listing price of the property. Hence, if the listing price would have changed, it would be a type 1 SCD, that stores the latest record.

e) A short explanation of the difference between the two versions of the star/snowflake schema.

The following changes have been made in the version-2 to remove aggregation:

- Host join time dimension has been replaced with host time dimension which contains host_since attribute.
- Season dimension has been replaced with property dimension which contains property details.
- Listing time dimension has been replaced with listing date dimension which contains the listing date.

With the help of above changes, a star schema with no aggregation (Level 0) has been created.

C.2 Implement version 1 star/snowflake schema using SQL

The version-1 of star/snowflake schema has been implemented in SQL. The below queries and outputs show the fact and dimension tables created in SQL.

1. Creating LISTING_TYPE Dimension

```
create table listing_type_dim as
select * from MSTAY.listing_type;
```

	TYPE_ID	TYPE_DESCRIPTION
1	1	Private room
2	2	Entire home/apt
3	3	Shared room
4	4	Hotel room

2. Creating LISTING_TIME Dimension

create table listing_time_dim as

select distinct

to_char(listing_date, 'YYYYMM') as listing_timeID,

to_char(listing_date, 'MM') as Month,

to_char(listing_date, 'YYYY') as Year

from listing;

	LISTING_TIMEID	MONTH	YEAR
1	201604	04	2016
2	201605	05	2016
3	201702	02	2017
4	201703	03	2017
5	201705	05	2017
6	201408	08	2014
7	201410	10	2014
8	201707	07	2017
9	201810	10	2018
10	201906	06	2019
11	201202	02	2012

3. Creating LISTING_SEASON Dimension

create table listing_season

(season varchar(10),

season_description varchar2(10));

insert into listing_season values ('Spring', 'Sep-Nov');

insert into listing_season values ('Summer', 'Dec-Feb');

insert into listing_season values ('Autumn', 'Mar-May');

insert into listing_season values ('Winter', 'Jun-Aug');

*select * from listing_season;*

	SEASON	SEASON_DESCRIPTION
1	Spring	Sep-Nov
2	Summer	Dec-Feb
3	Autumn	Mar-May
4	Winter	Jun-Aug

4. Creating LISTING_MAX_STAY Dimension

```
create table max_stay_dim
(duration varchar(10),
duration_description varchar2(30));

insert into max_stay_dim values ('short', 'less than 14 nights');
insert into max_stay_dim values ('medium', '14 to 30 nights');
insert into max_stay_dim values ('long', 'more than 30 nights');

select * from max_stay_dim;
```

	DURATION	DURATION_DESCRIPTION
1	short	less than 14 nights
2	medium	14 to 30 nights
3	long	more than 30 nights

5. Creating PROPERTY_SIZE Dimension

```
create table prop_size_dim
(prop_size varchar(10),
size_description varchar(50));

insert into prop_size_dim values ('small', 'minimum of 1 bed and 1 bedroom');
```

insert into prop_size_dim values ('medium', 'minimum of 3 beds and 2 bedrooms');

insert into prop_size_dim values ('large', 'more than 5 beds and more than 3 bedrooms');

*select * from prop_size_dim;*

	PROP_SIZE	SIZE_DESCRIPTION
1	small	minimum of 1 bed and 1 bedroom
2	medium	minimum of 3 beds and 2 bedrooms
3	large	more than 5 beds and more than 3 bedrooms

6. Creating RATING Dimension

create table rating_dim

(rating varchar(5),

rating_description varchar2(15));

insert into rating_dim values ('0-1', 'Poor');

insert into rating_dim values ('1-2', 'Not Good');

insert into rating_dim values ('2-3', 'Average');

insert into rating_dim values ('3-4', 'Good');

insert into rating_dim values ('4-5', 'Excellent');

*select * from rating_dim;*

	RATING	RATING_DESCRIPTION
1	1-star	0-1
2	2-star	1-2
3	3-star	2-3
4	4-star	3-4
5	5-star	4-5

7. Creating HOST_LOCATION Dimension

create table host_loc_dim as

select distinct host_location

from host;

*select * from host_loc_dim;*

	HOST_LOCATION
1	Elwood, Victoria, Australia
2	Aireys Inlet, Victoria, Australia
3	Healesville, Victoria, Australia
4	Olinda, Victoria, Australia
5	Camberwell, Victoria, Australia
6	New York, New York, United States
7	St Kilda, Victoria, Australia
8	Hawthorn East, Victoria, Australia
9	East Melbourne, Victoria, Australia
10	Newport Beach, California, United States
11	Flemington, Victoria, Australia
12	Shah Alam, Selangor, Malaysia

8. Creating HOST_JOIN_TIME dimension

create table host_join_time_dim as

select distinct

to_char(host_since, 'YYYYMM') as TimeID,

to_char(host_since, 'MM') as Month,

to_char(host_since, 'YYYY') as Year

from host;

*select * from host_join_time_dim;*

	TIMEID	MONTH	YEAR
1	201202	02	2012
2	201206	06	2012
3	201301	01	2013
4	201410	10	2014
5	201408	08	2014
6	201604	04	2016
7	201605	05	2016
8	201609	09	2016
9	201611	11	2016
10	201702	02	2017
11	201705	05	2017

9. Creating HOST dimension

```
create table host_dim as
select h.host_id,
    h.host_name,
    1.0/count(v.channel_id) as WeightFactor,
    LISTAGG (c.channel_name, '_') Within Group (Order By c.channel_name) as channel_list
from host h, host_verification v, mstay.channel c
where h.host_id = v.host_id
and v.channel_id = c.channel_id
group by h.host_id,h.host_name;

select * from host_dim;
```

	HOST_ID	HOST_NAME	WEIGHTFACTOR	CHANNEL_LIST
1	17308	Romi	0.11111111111111111111111111111111111111111	email_government_id_identity_manual_jumio_offline_government_id_phone_reviews_selfie_work_email
2	18785	Kate	0.2	email_facebook_jumio_phone_reviews
3	26687	Rachel	0.1	email_facebook_government_id_identity_manual_jumio_offline_government_id_phone_reviews_selfie_work_e
4	33057	Manju	0.33333333333333333333333333333333333333333	email_phone_reviews
5	40864	Jane	0.25	government_id_jumio_offline_government_id_reviews
6	50121	The A2C Team	0.14285714285714285714285714285714285714285	email_google_government_id_jumio_phone_reviews_work_email
7	94675	Ariel	0.16666666666666666666666666666666666666667	email_government_id_jumio_phone_reviews_work_email
8	98515	Aleksandra	0.125	email_facebook_government_id_identity_manual_offline_government_id_phone_reviews_selfie
9	112497	Fleur	0.125	email_government_id_identity_manual_jumio_offline_government_id_phone_reviews_selfie
10	123875	Ben	0.2	email_government_id_offline_government_id_phone_reviews
11	138184	Annie	0.2	email_jumio_offline_government_id_phone_reviews
12	138187	Julie	0.11111111111111111111111111111111111111111	email_facebook_government_id_identity_manual_jumio_offline_government_id_phone_reviews_selfie

10. Creating HOST_VERIFICATION_BRIDGE

create table host_verf_bridge as

*select * from host_verification;*

*select * from host_verf_bridge;*

	HOST_ID	CHANNEL_ID
1	1456169	4
2	1456169	8
3	1456169	5
4	1456169	10
5	6889546	1
6	6889546	2
7	6889546	3
8	12942356	1
9	12942356	2
10	12942356	3

11. Creating HOST_CHANNEL Dimension

create table host_channel_dim as

*select * from mstay.channel;*

*select * from host_channel_dim;*

	CHANNEL_ID	CHANNEL_NAME
1	1	email
2	2	phone
3	3	reviews
4	4	jumio
5	5	government_id
6	6	selfie
7	7	identity_manual
8	8	offline_government_id
9	9	facebook
10	10	work_email
11	11	manual_online

12. Creating LISTING Fact

```
drop table temp_listing_fact;
create table temp_listing_fact as
select
   l.listing_id,
   l.listing_price,
   l.listing_max_nights,
   l.listing_date,
   p.prop_num_beds,
   p.prop_num_bedrooms,
   p.prop_average_rating,
   t.type_id,
   h.host_location
from listing l, mstay.listing_type t, host h, mstay.property p
where l.type_id = t.type_id
and l.host_id = h.host_id
and l.prop_id = p.prop_id;

select * from temp_listing_fact;

alter table temp_listing_fact
add duration varchar(10);

alter table temp_listing_fact
add listing_time_id varchar(6);

alter table temp_listing_fact
add prop_size varchar(10);
```

```
alter table temp_listing_fact
add rating varchar(10);

alter table temp_listing_fact
add season varchar(10);

select * from temp_listing_fact;

update temp_listing_fact
set duration = 'long'
where listing_max_nights > 30;

update temp_listing_fact
set duration = 'short'
where listing_max_nights < 14;

update temp_listing_fact
set duration = 'medium'
where duration is null;

update temp_listing_fact
set listing_time_id = to_char(listing_date, 'YYYYMM');

update temp_listing_fact
set prop_size = 'medium'
where prop_num_beds <=5 and prop_num_beds >=3
and prop_num_bedrooms <=3 and prop_num_bedrooms >=2;

update temp_listing_fact
```

```
set prop_size = 'large'
where prop_num_beds >5
and prop_num_bedrooms >3;

update temp_listing_fact
set prop_size = 'small'
where prop_size is null;

update temp_listing_fact
set rating = '1-star'
where prop_average_rating > 0 and prop_average_rating <=1;

update temp_listing_fact
set rating = '2-star'
where prop_average_rating > 1 and prop_average_rating <=2;

update temp_listing_fact
set rating = '3-star'
where prop_average_rating > 2 and prop_average_rating <=3;

update temp_listing_fact
set rating = '4-star'
where prop_average_rating > 3 and prop_average_rating <=4;

update temp_listing_fact
set rating = '5-star'
where prop_average_rating > 4;

update temp_listing_fact
```

```
set season = 'Spring'
where to_char(listing_date, 'MM') in ('09','10','11');

update temp_listing_fact
set season = 'Summer'
where to_char(listing_date, 'MM') in ('12','01','02');

update temp_listing_fact
set season = 'Autumn'
where to_char(listing_date, 'MM') in ('03','04','05');

update temp_listing_fact
set season = 'Winter'
where to_char(listing_date, 'MM') in ('06','07','08');

create table listing_fact as
select
   listing_time_id,
   duration,
   prop_size,
   rating,
   season,
   host_location,
   type_id,
   count(*) as number_of_prop_listed,
   sum(listing_price) as total_listing_price
from temp_listing_fact
group by duration, listing_time_id, prop_size, rating, season, host_location, type_id;
```

*select * from listing_fact;*

	LISTING_TIME_ID	DURATION	PROP_SIZE	RATING	SEASON	HOST_LOCATION	TYPE_ID	NUMBER_OF_PROP_LISTED	TOTAL_LISTING_PRICE
1	201607	long	small	5-star	Winter	Fitzroy, Victoria, Australia	2	2	584
2	201708	long	small	5-star	Winter	Fitzroy, Victoria, Australia	2	2	584
3	201709	long	small	5-star	Spring	Fitzroy, Victoria, Australia	2	1	292
4	201412	medium	small	5-star	Summer	Melbourne, Victoria, Australia	2	5	560
5	201801	medium	small	5-star	Summer	Melbourne, Victoria, Australia	2	11	1122
6	201802	medium	small	5-star	Summer	Melbourne, Victoria, Australia	2	10	984
7	201106	medium	small	5-star	Winter	Victoria, Australia	2	3	450
8	201111	medium	small	5-star	Spring	Victoria, Australia	2	2	300
9	201202	medium	small	5-star	Summer	Victoria, Australia	2	2	300

13. Creating PROPERTY Fact

```
create table temp_prop_fact as
select prop_id,
    prop_average_rating,
    prop_num_beds,
    prop_num_bedrooms
from mstay.property;

alter table temp_prop_fact
add prop_size varchar(10);

alter table temp_prop_fact
add rating varchar(10);

update temp_prop_fact
set rating = '1-star'
where prop_average_rating > 0 and prop_average_rating <=1;

update temp_prop_fact
set rating = '2-star'
where prop_average_rating > 1 and prop_average_rating <=2;
```

```
update temp_prop_fact
set rating = '3-star'
where prop_average_rating > 2 and prop_average_rating <=3;

update temp_prop_fact
set rating = '4-star'
where prop_average_rating > 3 and prop_average_rating <=4;

update temp_prop_fact
set rating = '5-star'
where prop_average_rating > 4;

update temp_prop_fact
set prop_size = 'medium'
where prop_num_beds <=5 and prop_num_beds >=3
and prop_num_bedrooms <=3 and prop_num_bedrooms >=2;

update temp_prop_fact
set prop_size = 'large'
where prop_num_beds >5
and prop_num_bedrooms >3;

update temp_prop_fact
set prop_size = 'small'
where prop_size is null;

create table property_fact as
select
```

```
    prop_size,
    rating,
    count(*) as num_of_properties
from temp_prop_fact
group by prop_size, rating;

select * from property_fact;
```

	PROP_SIZE	RATING	NUM_OF_PROPERTIES
1	small	3-star	27
2	small	1-star	14
3	small	2-star	9
4	medium	4-star	21
5	medium	5-star	528
6	large	5-star	3
7	small	4-star	182
8	medium	3-star	1
9	small	5-star	3379

14. Creating HOST Fact

```
create table temp_host_fact as
select
    host_id,
    host_since,
    host_location
from host;

alter table temp_host_fact
add host_join_time_id varchar(6);

update temp_host_fact
set host_join_time_id = to_char(host_since, 'YYYYMM');
```

```
create table host_fact as
select
   host_id,
   host_join_time_id,
   host_location,
   count(*) as num_of_hosts
from temp_host_fact
group by host_id, host_join_time_id, host_location;
```

```
select * from host_fact;
```

	HOST_ID	HOST_JOIN_TIME_ID	HOST_LOCATION
1	2716860	201206	Melbourne, Victoria, Australia
2	3721646	201210	Elwood, Victoria, Australia
3	2303218	201205	Melbourne, Victoria, Australia
4	5370551	201303	Diamond Creek, Victoria, Australia
5	6432048	201305	Melbourne, Victoria, Australia
6	7613976	201307	Chum Creek, Victoria, Australia
7	2692055	201206	Melbourne, Victoria, Australia
8	13397269	201403	Victoria, Australia
9	22420759	201410	Melbourne, Victoria, Australia
10	30440408	201504	Melbourne, Victoria, Australia
11	33401062	201505	Melbourne, Victoria, Australia
12	6354616	201305	Melbourne, Victoria, Australia

C.3 Create the following reports using OLAP queries.

a. Simple reports:

Report 1

(a) Query Question

Q: Show the Top 5 years when the maximum number of hosts registered for each location.

(b) Importance of query

This query would help the management to observe the trends of host joining across the years.

(c) The SQL commands

```
select * from
(select
    host_location,
    year as join_year,
    sum(num_of_hosts) as num_of_hosts,
    RANK() OVER (PARTITION BY host_location ORDER BY sum(num_of_hosts) DESC) AS RANK_BY_LOCATION
from host_fact h, host_join_time_dim t
where h.host_join_time_id = t.timeid
group by host_location, year)
where rank_by_location<=5;
```

(d) The screenshots of the query results

	HOST_LOCATION	JOIN_YEAR	NUM_OF_HOSTS	RANK_BY_LOCATION
1	AU	2018	79	1
2	AU	2015	75	2
3	AU	2016	62	3
4	AU	2017	54	4
5	AU	2014	25	5
6	AUSTRALIA	2014	1	1
7	Abbotsford, Victoria, Australia	2015	5	1
8	Abbotsford, Victoria, Australia	2014	5	1
9	Abbotsford, Victoria, Australia	2011	2	3
10	Abbotsford, Victoria, Australia	2016	2	3
11	Abbotsford, Victoria, Australia	2013	1	5
12	Abbotsford, Victoria, Australia	2012	1	5
13	Abbotsford, Victoria, Australia	2018	1	5

Report 2

(a) Query Question

Q: Show the years where Top 30% of properties are listed by season.

(b) Importance of query

This query would help the management to observe in which years the maximum properties were listed based on season.

(c) The SQL commands

```
select *
from
(
select
   season,
   year,
   sum(number_of_prop_listed) as number_of_prop_listed,
   PERCENT_RANK() OVER (PARTITION BY season ORDER BY sum(number_of_prop_listed) DESC) AS RANK_BY_SEASON
from listing_fact l, listing_time_dim t
where t.listing_timeid = l.listing_time_id
group by season,year)
where rank_by_season>0.7;
```

(d) The screenshots of the query results

	SEASON	YEAR	NUMBER_OF_PROP_LISTED	RANK_BY_SEASON
1	Autumn	2020	41	0.8
2	Autumn	2012	37	0.9
3	Autumn	2011	3	1
4	Spring	2020	62	0.72727272727272727272727272727272727273
5	Spring	2011	23	0.81818181818181818181818181818181818182
6	Spring	2021	5	0.90909090909090909090909090909090909091
7	Spring	2010	2	1
8	Summer	2012	61	0.8
9	Summer	2021	59	0.9
10	Summer	2011	13	1

Report 3

(a) Query Question

Q: What are the subtotals and total listing price from each listing type, season, and listing duration?

(b) Importance of query

This query would help the management to observe the subtotals of total listing price from different perspective such as listing type, season, and listing duration.

(c) The SQL commands

```
select DECODE(GROUPING(t.type_description), 1, 'All listing types', t.type_description) AS listing_type,
    DECODE(GROUPING(season), 1, 'All seasons', season) AS season,
    DECODE(GROUPING(duration), 1, 'All durations', duration) AS duration,
    sum(total_listing_price) as total_listing_price
from listing_fact l, listing_type_dim t
where l.type_id = t.type_id
group by cube(t.type_description, season, duration);
```

(d) The screenshots of the query results

	LISTING_TYPE	SEASON	DURATION	TOTAL_LISTING_PRICE
1	All listing types	All seasons	All durations	609723
2	All listing types	All seasons	long	488877
3	All listing types	All seasons	short	1127
4	All listing types	All seasons	medium	119719
5	All listing types	Autumn	All durations	142196
6	All listing types	Autumn	long	114522
7	All listing types	Autumn	short	343
8	All listing types	Autumn	medium	27331
9	All listing types	Spring	All durations	161409
10	All listing types	Spring	long	129322
11	All listing types	Spring	short	441
12	All listing types	Spring	medium	31646

Report 4

(a) Query Question

Q: What are the subtotals and total listing price from each listing type and listing duration for each season?

(b) Importance of query

This query would help the management to observe the subtotals of total listing price based on listing type and listing duration for every season.

(c) The SQL commands

```
select DECODE(GROUPING(season), 1, 'All seasons', season) AS season,
    DECODE(GROUPING(t.type_description), 1, 'All listing types', t.type_description) AS listing_type,
    DECODE(GROUPING(duration), 1, 'All durations', duration) AS duration,
    sum(total_listing_price) as total_listing_price
from listing_fact l, listing_type_dim t
where l.type_id = t.type_id
group by season, cube(t.type_description, duration);
```

(d) The screenshots of the query results

	SEASON	LISTING_TYPE	DURATION	TOTAL_LISTING_PRICE
1	Autumn	All listing types	All durations	142196
2	Autumn	All listing types	long	114522
3	Autumn	All listing types	short	343
4	Autumn	All listing types	medium	27331
5	Autumn	Private room	All durations	903
6	Autumn	Private room	long	560
7	Autumn	Private room	short	343
8	Autumn	Entire home/apt	All durations	141293
9	Autumn	Entire home/apt	long	113962
10	Autumn	Entire home/apt	medium	27331
11	Spring	All listing types	All durations	161409
12	Spring	All listing types	long	129322

Report 5

(a) Query Question

Q: What are the subtotals and total listing price from each property size and rating?

(b) Importance of query

This query would help the management to observe the subtotals of total listing price based on property size and rating.

(c) The SQL commands

```
select
   DECODE(GROUPING(prop_size), 1, 'All sizes', prop_size) AS prop_size,
   DECODE(GROUPING(rating), 1, 'All ratings', rating) AS rating,
   sum(num_of_properties) as total_listing_price
from property_fact
group by rollup(prop_size,rating);
```

(d) The screenshots of the query results

	PROP_SIZE	RATING	TOTAL_LISTING_PRICE
1	large	5-star	3
2	large	All ratings	3
3	small	1-star	14
4	small	2-star	9
5	small	3-star	27
6	small	4-star	182
7	small	5-star	3379
8	small	All ratings	3611
9	medium	3-star	1
10	medium	4-star	21
11	medium	5-star	528
12	medium	All ratings	550

Report 6

(a) Query Question

Q: What are the subtotals and total listing price from season and property size for each year?

(b) Importance of query

This query would help the management to observe the subtotals of total listing price based on property size and season for each year.

(c) The SQL commands

```
select
   year AS listing_year,
   DECODE(GROUPING(prop_size), 1, 'All sizes', prop_size) AS prop_size,
   DECODE(GROUPING(season), 1, 'All seasons', season) AS season,
   sum(total_listing_price) as total_listing_price
from listing_fact l,listing_time_dim t
where l.listing_time_id = t.listing_timeid
group by year, rollup(prop_size,season);
```

(d) The screenshots of the query results

	LISTING_YEAR	PROP_SIZE	SEASON	TOTAL_LISTING_PRICE
1	2010	small	Winter	95
2	2010	small	All seasons	95
3	2010	medium	Spring	198
4	2010	medium	All seasons	198
5	2010	All sizes	All seasons	293
6	2011	small	Autumn	210
7	2011	small	Spring	2583
8	2011	small	Summer	1259
9	2011	small	Winter	1534
10	2011	small	All seasons	5586

Report 7

(a) Query Question

Q: What are the total listing price and cumulative total listing price of small properties in each year?

(b) Importance of query

This query will help the management to analyze the trend of listing price of small properties in every year.

(c) The SQL commands

```
select
    year as listing_year,
    SUM(total_listing_price) as total_listing_price,
    SUM(SUM(total_listing_price)) OVER (ORDER BY year ROWS UNBOUNDED PRECEDING) AS CUM_LISTING_PRICE
from listing_fact l, listing_time_dim t
where l.listing_time_id = t.listing_timeid
and prop_size = 'small'
group by year;
```

(d) The screenshots of the query results

	LISTING_YEAR	TOTAL_LISTING_PRICE	CUM_LISTING_PRICE
1	2010	95	95
2	2011	5586	5681
3	2012	22760	28441
4	2013	38286	66727
5	2014	53336	120063
6	2015	62805	182868
7	2016	70169	253037
8	2017	68891	321928
9	2018	64543	386471
10	2019	60310	446781

Report 8

(a) Query Question

Q: What is the total number of hosts joining and 3 years moving average of host joining every year?

(b) Importance of query

This query will help the management to analyze how many hosts are joining every year.

(c) The SQL commands

```
select
   year as host_join_year,
   SUM(h.num_of_hosts) as num_of_hosts,
   ROUND(AVG(SUM(h.num_of_hosts)) OVER (ORDER BY year ROWS 2 PRECEDING),2) AS MOVING_3_MON_AVG
from host_fact h, host_join_time_dim t
where h.host_join_time_id = t.timeid
group by year;
```

(d) The screenshots of the query results

	HOST_JOIN_YEAR	NUM_OF_HOSTS	MOVING_3_MON_AVG
1	2009	6	6
2	2010	31	18.5
3	2011	122	53
4	2012	288	147
5	2013	444	284.67
6	2014	631	454.33
7	2015	921	665.33
8	2016	710	754
9	2017	401	677.33
10	2018	272	461
11	2019	38	237
12	2020	8	106

Report 9

(a) Query Question

Q: Show ranking of each property size and ranking of each property rating based on the total number of properties.

(b) Importance of query

This query will help the management to know the highest numbers of properties listed based on property size and ratings.

(c) The SQL commands

```
select
   prop_size,
   rating,
   sum(num_of_properties),
   RANK() OVER (PARTITION BY prop_size ORDER BY SUM(num_of_properties) DESC) AS RANK_BY_SIZE,
   RANK() OVER (PARTITION BY rating ORDER BY SUM(num_of_properties) DESC) AS RANK_BY_RATING
from property_fact
group by prop_size, rating;
```

(d) The screenshots of the query results

	PROP_SIZE	RATING	SUM(NUM_OF_PROPERTIES)	RANK_BY_SIZE	RANK_BY_RATING
1	large	5-star	3	1	3
2	medium	5-star	528	1	2
3	medium	4-star	21	2	2
4	medium	3-star	1	3	2
5	small	5-star	3379	1	1
6	small	4-star	182	2	1
7	small	3-star	27	3	1
8	small	1-star	14	4	1
9	small	2-star	9	5	1

Report 10

(a) Query Question

Q: Show ranking of each listing type and ranking of each season based on the average listing price.

(b) Importance of query

This query will help the management to know the average prices of properties ranked on the basis of season and property type.

(c) The SQL commands

```
select
   type_description as listing_type,
   season,
   round(sum(total_listing_price)/sum(number_of_prop_listed),2) as average_listing_price,
   RANK() OVER (PARTITION BY type_description ORDER BY sum(total_listing_price)/sum(number_of_prop_listed) DESC) AS RANK_BY_LISTING_TYPE,
   RANK() OVER (PARTITION BY season ORDER BY sum(total_listing_price)/sum(number_of_prop_listed) DESC) AS RANK_BY_SEASON
from listing_fact l, listing_type_dim t
where l.type_id = t.type_id
group by type_description, season;
```

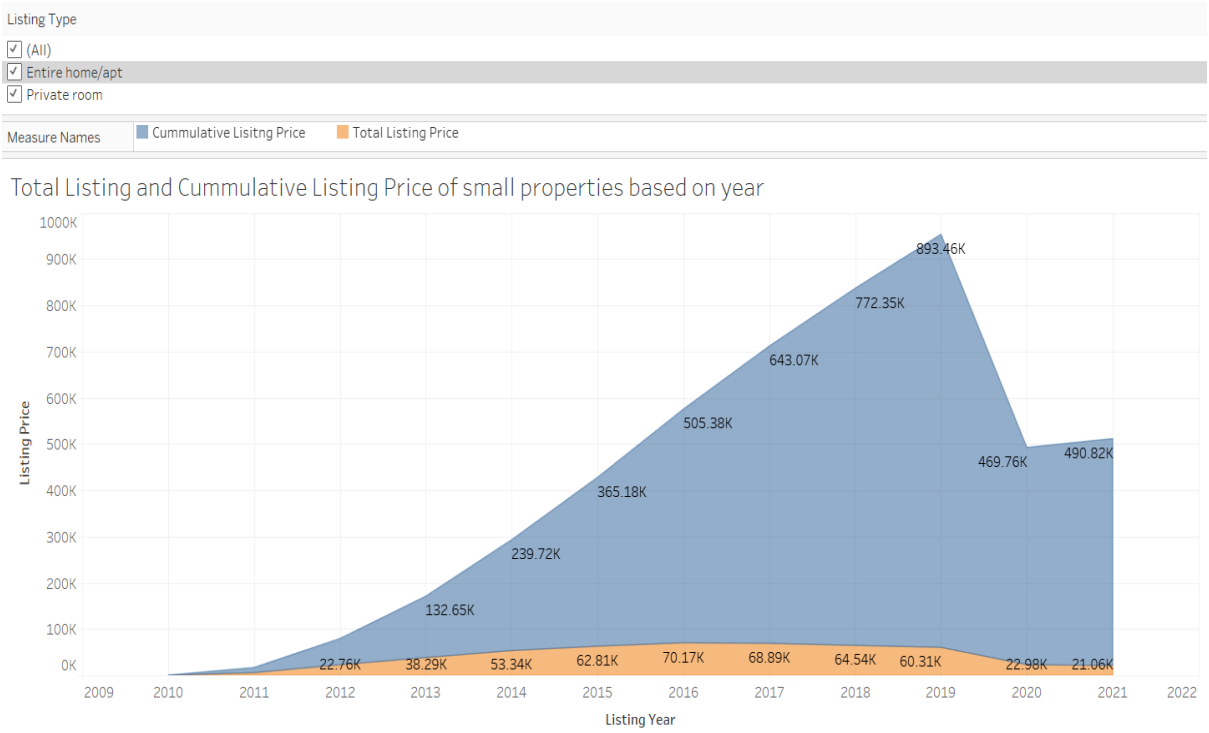
(d) The screenshots of the query results

	LISTING_TYPE	SEASON	AVERAGE_LISTING_PRICE	RANK_BY_LISTING_TYPE	RANK_BY_SEASON
1	Entire home/apt	Summer	125.39	1	1
2	Entire home/apt	Winter	124.34	2	1
3	Entire home/apt	Spring	124.22	3	1
4	Entire home/apt	Autumn	122.33	4	1
5	Private room	Summer	93.25	1	2
6	Private room	Autumn	69.46	2	2
7	Private room	Winter	67.22	3	2
8	Private room	Spring	60.85	4	2

C.4. Business Intelligence (BI) Reports

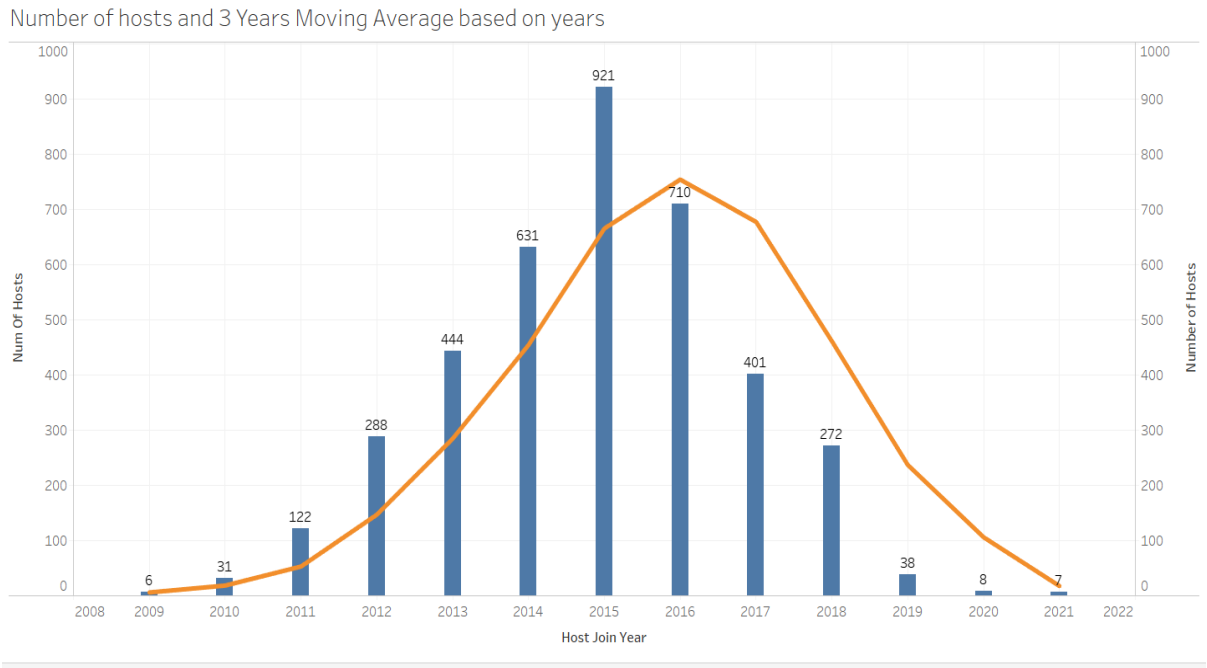
The following reports have been chosen from section C.3

Report 7: What are the total listing price and cumulative total listing price of small properties in each year?



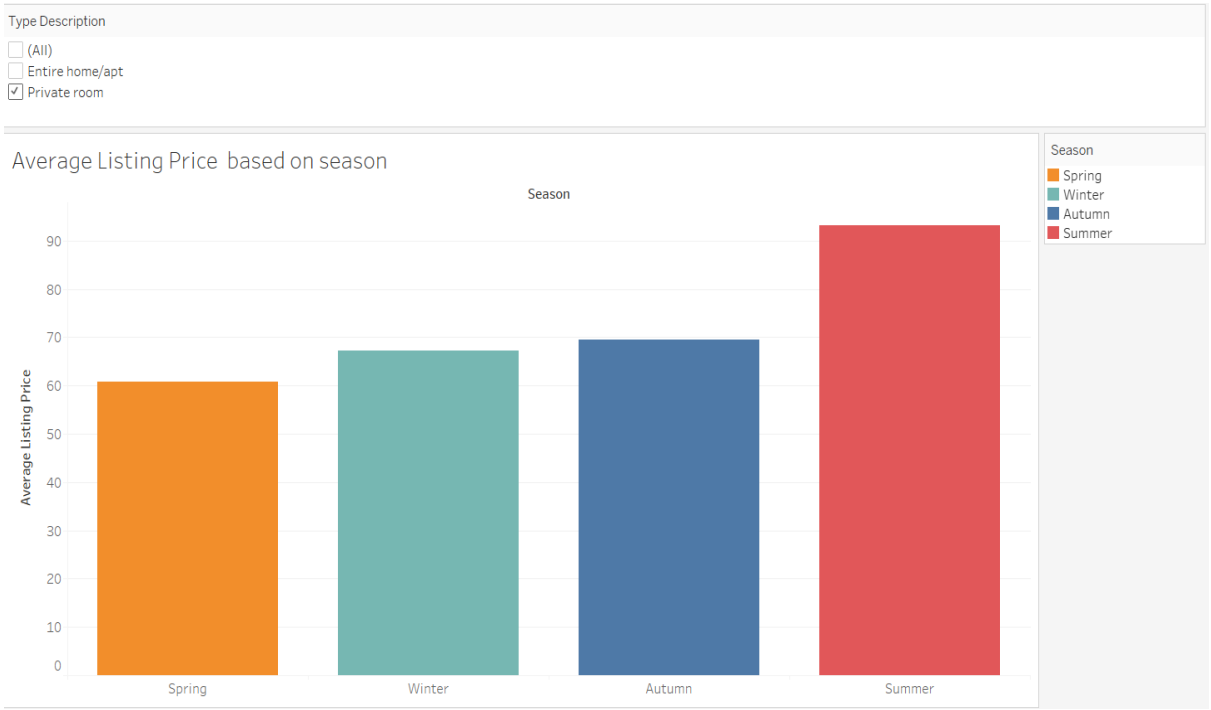
The above figure shows the area chart depicting total listing price and cumulative listing price over the years. The dashboard also gives user the feature to select type of property for the analysis.

Report 8: What is the total number of hosts joining and 3 years moving average of host joining every year?



The above figure shows the dashboard depicting bar graph of number of hosts along with 3 years moving average of number of hosts joining based on years.

Report 10: Show ranking of each listing type and ranking of each season based on the average listing price.



The above figure shows the dashboard depicting the rank of season based on average listing price. The dashboard also gives user the feature to select type of property for the analysis.